Our goal today is to show that Exact Triangle and Sparse Triangle Listing are hard under both 3SUM and APSP.

Let's define the two problems:

The **Exact Triangle** problem (also known as Zero Triangle) is as follows: Given a tripartite graph $G = (V, E)$ with node parts A, B, C of size n each and edge weights in $[\pm U]$ where U has poly$(\log n)$ bits, are there $a \in A, b \in B, c \in C$ so that $w(a, b) + w(b, c) + w(a, c) = 0$?

> **Exercise**: Convince yourselves that Exact Triangle is equivalent to the following two other versions.
>
> - Given a not necessarily tripartite graph G with weights in $[\pm U]$ where U has poly$(\log n)$ bits, determine whether G contains a triangle whose edge weight sum to 0.
> - Given a tripartite graph $G = (V, E)$ with node parts A, B, C of size n each and edge weights in $[\pm U]$ where U has poly$(\log n)$ bits, determine whether there are $a \in A, b \in B, c \in C$ so that $w(a, b) + w(b, c) = w(a, c)$.

The **Sparse Triangle Listing** problem is as follows: Given an m edge graph G, list m triangles in G and if G has $< m$ triangles, list all of its triangles.

The brute-force algorithm solves Exact Triangle in $O(n^3)$ time. We will show that an $O(n^{3-\varepsilon})$ time algorithm for $\varepsilon > 0$ for it would refute both the 3SUM and the APSP Hypotheses.

If $\omega = 2$, the fastest known algorithm for Sparse Triangle Listing runs in $\tilde{O}(m^{4/3})$ time. We will show that an $O(m^{4/3-\varepsilon})$ time algorithm for $\varepsilon > 0$ would imply an $O(n^{3-\delta})$ time algorithm for $\delta > 0$ for Exact Triangle and hence refute both the APSP and 3SUM Hypotheses.

1 APSP to Exact Triangle– your next problem set

We showed that APSP in n node graphs with weights in $[\pm U]$ where U has poly$(\log n)$ bits is equivalent to the **Negative Triangle** problem: Given a tripartite graph $G = (V, E)$ with node parts A, B, C of size n each and edge weights in $[\pm U]$ where U has poly$(\log n)$ bits, are there $a \in A, b \in B, c \in C$ so that $w(a, b) + w(b, c) + w(a, c) < 0$?

In your next problem set, you will show how to reduce Negative Triangle to poly$(\log n)$ calls to Exact Triangle.

Here's the setup:

For a nonnegative integer w on B bits and $i \in [B]$, let $pre_i(x)$ be the integer consisting of the first i bits in the bit representation of x (the prefix of the first i bits). E.g. if $x = 001100$, $pre_4(x) = 0011$.

Claim 1. *If a and b have B bits with leading bit 0 and c has B bits, then $a + b > c$ if and only if either*

- *there exists some $i \in [B]$ for which $pre_i(a) + pre_i(b) = pre_i(c) + 1$, or*
- *there exists some $i \in [B]$ for which $pre_i(a) + pre_i(b) = pre_i(c)$ and $pre_{i+1}(a) = pre_i(a)1$, $pre_{i+1}(b) = pre_i(b)1$ and $pre_{i+1}(c) = pre_i(c)0$.*

> Exercise: Prove the above claim!

Corollary 1.1. *Negative Triangle in tripartite graphs with poly$\log(n)$ bit weights can be reduced to poly$\log(n)$ instances of Exact Triangle with poly$\log(n)$ bit weights.*

Exercise: Prove the above corollary!

As a corollary we get that if Exact Triangle is in $O(n^{3-\varepsilon})$ time for $\varepsilon > 0$, then APSP is in $O(n^{3-\delta})$ time for $\delta > 0$.

2 3SUM to Exact Triangle

We will focus on reductions from 3SUM-Convolution. Recall that the 3SUM-Convolution problem is, given an integer array A of length n, are there i, j, $i \neq j$ so that $A[i] + A[j] = A[i + j]$?

Last time we showed that 3SUM and 3SUM-Convolution are subquadratically equivalent. Now we reduce 3SUM-Convolution to Exact Triangle.

In fact the reduction is a generic (n^2, n^3)-fine-grained reduction from any convolution problem to its triangle version.

Definition 2.1 (f-Convolution, f-Triangle)**.** *Let $f : \mathbb{Z}^3 \to \{0, 1\}$ be any function that takes three integers x, y, z to a boolean value $f(x, y, z) \in \{0, 1\}$. Then*

- *The* f-Convolution *problem is: given an integer array X of length n, determine if there are $j, i \in [n]$ j so that $f(X[i], X[j - i], X[j]) = 1$.*
- *The* f-Triangle *problem is: given a tripartite graph on parts A, B, C on n nodes each and integer edge weights, determine if there are $a \in A, b \in B, c \in C$ with $f(w(a, b), w(b, c), w(a, c)) = 1$.*

Theorem 2.1. *For every f, there is an $O(n^{1.5})$ time reduction from f-Convolution to $\sqrt{n}$ oracle calls to f-Triangle on $\sqrt{n}$ node graphs.*

Theorem 2.1 shows that if f-Triangle can be solved in $O(n^{3-\varepsilon})$ time for $\varepsilon > 0$ in n node graphs, then f-Convolution on n length sequences can be solved in $O(\sqrt{n} \times (\sqrt{n})^{3-\varepsilon}) = O(n^{2-\varepsilon/2})$ time.

Using the function $f(x, y, z) = 1$ if $x + y = z$ and $f(x, y, z) = 0$ otherwise, and using the equivalent version of Exact Triangle in which we want a triangle a, b, c with $w(a, b) + w(b, c) = w(a, c)$, we obtain:

Corollary 2.1. *There is an $O(n^{1.5})$ time reduction from 3SUM Convolution to $\sqrt{n}$ oracle calls to Exact Triangle on $\sqrt{n}$ node graphs.*

Hence if Exact Triangle on N node graphs has an $O(N^{3-\varepsilon})$ time algorithm, then 3SUM Convolution can be solved in $O(n^{2-\varepsilon/2})$ time.

Proof of Theorem 2.1. Suppose we are given an instance of f-Convolution, an array X of length n, $[X[0], \ldots, X[n - 1]]$.

For every index $i \in \{0, \ldots, \sqrt{n} - 1\}$, we create a graph G_i as follows. G_i is tripartite with partitions U_i, V_i, W_i. U_i contains a node t for every $t \in \{0, \ldots, \sqrt{n}-1\}$, W_i contains a node q for every $q \in \{0, \ldots, 2\sqrt{n}-2\}$, and V_i contains a node s for every $s \in \{0, \ldots, \sqrt{n} - 1\}$.

The edges of G_i are as follows.

- For every $q \in W_i, t \in U_i$, if $q - t \in \{0, \ldots, \sqrt{n} - 1\}$, we add an edge (q, t) with weight $X[i\sqrt{n} + (q - t)]$.
- For every $s \in V_i, t \in U_i$, add an edge (s, t) with weight $X[s\sqrt{n} + t]$.
- For every $s \in V_i, q \in W_i$, add an edge (s, q) with weight $X[(s + i)\sqrt{n} + q]$.

Claim 2. *X has a f-Convolution solution if and only if for some $i \in \{0, \ldots, \sqrt{n} - 1\}$, G_i contains an f-Triangle.*

Proof. Below we will assume that the indices never overflow. Suppose that X has a f-Convolution: some k, j, $k \neq j$ such that $f(X[k], X[j], X[k+j]) = 1$. Now, $k = i\sqrt{n} + \ell$ for some $i, \ell \in \{0, \ldots, \sqrt{n}-1\}$, and $j = s\sqrt{n} + t$ for some $s, t \in \{0, \ldots, \sqrt{n}-1\}$. Thus also, $k + j = (s+i)\sqrt{n} + (t+\ell)$.

Consider graph G_i. The nodes $t \in U_i, s \in V_i, (t+\ell) \in W_i$ in G_i form a triangle (since $(t+\ell) - t \in \{0, \ldots, \sqrt{n}-1\}$). The f-weight of this triangle is

$$f(X[s\sqrt{n}+t], X[i\sqrt{n}+\ell], X[(s+i)\sqrt{n}+(t+\ell)]) = f(X[j], X[k], X[k+j]) = 1.$$

Thus G_i has an f-Triangle.

Now, let us assume that for some i, G_i contains an f-Triangle ($q \in W_i, s \in V_i, t \in U_i$). The f-weight of the triangle is $f(X[s\sqrt{n}+t], X[i\sqrt{n}+(q-t)], X[(s+i)\sqrt{n}+q]) = 1$.

Let $a = s\sqrt{n}+t, b = i\sqrt{n}+(q-t), c = (s+i)\sqrt{n}+q$. Notice that $c = a + b$. Also, by the above, $f(X[a], X[b], X[c]) = 1$, and we have a f-Convolution solution.

□

The first proof of this reduction as far as we know appears in [3]. The reduction there is not presented as a general but only in the context in reducing the so called $(\min, +)$-Convolution to $(\min, +)$-Product. The reduction appears again in other contexts, for instance in reducing 3-SUM Convolution to Exact Triangle [6].

Let's briefly give the application of the theorem to $(\min, +)$-Convolution. The $(\min, +)$-Convolution problem is, given arrays A and B of length n, compute an array C s.t. for all j, $C[j] = \min_i A[i] + B[j-i]$ (assuming no index overflows as before).

> **Exercise:** Show that $(\min, +)$-Convolution is (n^2, n^2)-fine-grained equivalent to *Negative Convolution*: Given three arrays A, B, C, determine if there are i, j s.t. $A[i] + B[j] + C[j-i] < 0$.

Now, let's use the function $f(x, y, z) = 1$ when $x + y + z < 0$ and $f(x, y, z) = 0$ otherwise. Using our general theorem and this f, we immediately obtain that Negative Convolution can be (n^2, n^3)-reduced to Negative Triangle, and thus $(\min, +)$-Convolution can be (n^2, n^3)-reduced to APSP.

3 Exact Triangle to Sparse Triangle Listing

We now want to reduce Exact Triangle to Sparse Triangle Listing. We will do this in two steps: (1) Exact Triangle Weight Reduction, (2) Exact Triangle Listing with Small Edge Weights to Sparse Triangle Listing.

This section of the notes is based on the proof by Chan and Xu in SOSA [4]. The original proof is by Vassilevska W. and Xu [7]. Prior to that, Patrascu [5] showed that 3SUM can be $(n^2, m^{4/3})$-reduced to Sparse Triangle Listing but no reduction from APSP was known.

3.1 Weight Reduction for Exact Triangle

This section is very similar to last lecture's weight reductions. Let's assume that we are given an Exact Triangle instance G with node partitions A, B, C of n nodes each and edge weights in $[\pm 2^m]$ where m is polylog(n).

Let's pick a random prime p from the interval $[n^3/(2t), n^3/t]$ for some parameter t, a small polynomial in n. Then, for any $x, y, z \in [\pm 2^m]$ with $x + y + z \neq 0$, there are $O(m)$ prime divisors of $x + y + z$. Also, there are $\Theta(n^3/(t \log n))$ primes in the interval $[n^3/(2t), n^3/t]$. Thus,

$$Pr_p[x + y + z = 0 \mod p] \leq O((mt \log n)/n^3) = \tilde{O}(t/n^3).$$

Thus if we take G and replace every edge weight with it mod p, obtaining a graph G_p we get that if G has L exact triangles, then the expected number of Exact Triangles mod p in G_p is $L + \tilde{O}(t)$. By Markov's

inequality, with probability $\geq 2/3$, the number of exact triangles mod p in G_p is at most $L + Q$ for some $Q \leq \tilde{O}(t)$. Also, if the number of exact triangles mod p in G_p is at most $L + Q$ and we find $Q + 1$ exact triangles mod p in G_p, then at least one of the triangles we list must be a true exact triangle in G.

Thus, we can solve Exact Triangle as follows:

Repeat $10 \log_3 n$ times:
  Pick a random prime p in $[n^3/(2t), n^3/t]$
  Form G_p
  List up to $Q + 1$ Exact triangles (mod p) in G_p
  If one of these triangles is a real exact triangle in G, return YES.
At the end return NO.

If G does not have an exact triangle, then the algorithm will always return NO.

If in any iteration the number of Exact triangles mod p listed is $< Q + 1$, this means that ALL exact triangles mod p were listed in that iteration. Thus, if no true exact triangle is found in that iteration, then G has no exact triangle.

The remaining case is that in all $10 \log_3 n$ iterations at least $Q + 1$ exact triangles mod p were found. We have that if G has an exact triangle, the probability that we get $\geq Q + 1$ false exact triangles in all $10 \log_3 n$ iterations is at most $(1/3)^{10 \log_3 n} = 1/n^{10}$ (since this means that G_p has at least $Q + 1 + L$ exact triangles mod p).

Combining the above two paragraphs: if G has an exact triangle, then with probability $\geq 1 - 1/n^{10}$ we will find one.

We have thus reduced Exact Triangle to Listing up to $\tilde{O}(t)$ Exact Triangles in a graph with weights bounded by n^3/t. (Recall that Exact Triangle mod p can be reduced to $O(1)$ instances of Exact Triangle, without the mod.)

3.2 Listing Exact Triangles to Sparse Triangle Listing

From above and via reweighting a bit, we get that Exact Triangle on n nodes can be reduced to the following problem: Given a tripartite graph G with node parts A, B, C each of size n and edge weights in $[W]$, where $W = \tilde{O}(n^3/t)$, if G contains at most t triangles $a \in A, b \in B, c \in C$ with $w(a, b) + w(b, c) = w(a, c)$, list all such triangles in G, and otherwise if it has more than t such triangles, list an arbitrary subset of t of them. (We call this "listing up to t" exact triangles.)

Now we reduce this problem to Listing m triangles in an $O(m)$-edge graph.

From Exact Triangle to 3D-Vector Exact triangle. For an integer $x \in [W]$ we can represent it uniquely by $x = x_1 \cdot q^2 + x_2 \cdot q + x_3$ for integer $q = \Theta(W^{1/3})$ and $x_1, x_2, x_3 \in \{0, \ldots, q - 1\}$. That is, we consider the representation of x in q-ary.

Now, $x + y = z$ if and only if $(x_1, x_2, x_3) + (y_1, y_2, y_3) = (z_1 - c_1, z_2 - c_2 + c_1 q, z_3 + c_2 q)$ for some choice $c_1, c_2 \in \{0, 1\}$; the c_i are just the carries when adding x and y in their q-ary representation.

Thus, for all 4 choices of $c_1, c_2 \in \{0, 1\}$ we can reduce our Exact Triangle listing problem to 4 instances of the following **3D-Vector Exact triangle listing** problem:

Given a tripartite graph G with node parts A, B, C each of size n where each edge (u, v) has a weight vector $\vec{w}(u, v) := (w_1(u, v), w_2(u, v), w_3(u, v)) \in [q]^3$, where $q = \Theta(W^{1/3}) = \tilde{O}(n/t^{1/3})$, if G contains at most t triangles $a \in A, b \in B, c \in C$ with $\vec{w}(a, b) + \vec{w}(b, c) = \vec{w}(a, c)$ (vector sum!), list all such triangles in G, and otherwise if it has more than t such triangles, list an arbitrary subset of t of them.

In particular, for every choice $c_1, c_2 \in \{0, 1\}$, create a graph G_{c_1, c_2} on the same vertex set as G and for every $a \in A, b \in B, c \in C$,

- if (a, b) is an edge of weight $w(a, b) = x_1 \cdot q^2 + x_2 \cdot q + x_3$, add an edge (a, b) to G_{c_1, c_2} with weight vector (x_1, x_2, x_3),

- if (b,c) is an edge of weight $w(b,c) = y_1 \cdot q^2 + y_2 \cdot q + y_3$, add an edge (b,c) to G_{c_1,c_2} with weight vector (y_1, y_2, y_3), and
- if (a,c) is an edge of weight $w(a,c) = z_1 \cdot q^2 + z_2 \cdot q + z_3$, add an edge (a,c) to G_{c_1,c_2} with weight vector $(z_1 - c_1, z_2 - c_2 + c_1 q, z_3 + c_2 q)$.

We want to list up to t exact vector triangles in all these instances.

From 3D-Vector Exact triangle listing to Sparse Triangle Listing. Now we are given one of these 3D-Vector Exact triangle listing instances on a tripartite graph with n node parts A, B, C where the weight vectors are in $[q]^3$ where $q \leq \tilde{O}(n/t^{1/3})$ and we want to list up to t triangles a, b, c with $\vec{w}(a,b) + \vec{w}(b,c) = \vec{w}(a,c)$.

We create a new unweighted tripartite graph H. The node parts of H are $A \times [q]^2, B \times [q]^2, C \times [q]^2$. We add edges as follows:

- For every $a \in A, b \in B$ with vector weight (x_1, x_2, x_3) we add an edge between (a, x_1, z_2) and (b, y_2, x_3) for every y_2 and z_2 such that $x_2 + y_2 = z_2$.
- For every $b \in B, c \in C$ with vector weight (y_1, y_2, y_3) we add an edge between (b, y_2, x_3) and (c, z_3, y_1) for every x_3 and z_3 such that $x_3 + y_3 = z_3$.
- For every $a \in A, c \in c$ with vector weight (z_1, z_2, z_3) we add an edge between (a, x_1, z_2) and (c, z_3, y_1) for every x_1 and y_1 such that $x_1 + y_1 = z_1$.

Notice that the procedure takes $O(n^2 q) = O(n^2 W^{1/3}) = \tilde{O}(n^3/t^{1/3})$ time and the number of edges is $O(n^2 q)$. For every triangle (a,b,c) in the original graph G with weights $\vec{w}(a,b) = (x_1, x_2, x_3), \vec{w}(b,c) = (y_1, y_2, y_3), \vec{w}(a,c) = (z_1, z_2, z_3)$ with $x_1 + y_1 = z_1, x_2 + y_2 = z_2, x_3 + y_3 = z_3$, there is a *unique* triangle in H given by $(a, x_1, z_2), (b, y_2, x_3), (c, z_3, y_1)$. I.e. every triangle in H is in one to one correspondence with every vector exact triangle of G. Thus listing up to t triangles in H corresponds to listing up to t triangles in G.

We have reduced Exact Triangle to Sparse Triangle Listing on a graph with $m = O(n^2 q) = \tilde{O}(n^3/t^{1/3})$ edges and where we want to list t triangles.

We set $t = n^3/t^{1/3}$ and thus $t = n^{9/4}$ and we have reduced the problem to listing up to $m = n^{9/4}$ triangles in an $\tilde{O}(m)$ edge unweighted graph.

If this latter problem can be solved in $O(m^{4/3-\varepsilon})$ time for some $\varepsilon > 0$, then we can solve Exact Triangle in time, up to polylogs,

$$(n^{9/4})^{4/3-\varepsilon} = n^{3-9\varepsilon/4}.$$

In fact, suppose that for *some* choice of $t = m^\delta$ for $\delta < 3/2$ one can list t triangles in an $m = n^3/t^{1/3}$ edge graph in $O((mt^{1/3})^{1-\varepsilon})$ time for some $\varepsilon > 0$. Then in our reduction, $t = m^\delta = (n^3/t^{1/3})^\delta$, so that $t = n^{3-3\delta/(3+\delta)}$. For $\delta' = 3\delta/(3+\delta)$, $t = n^{3-\delta'}$.

Notice that $\delta' < 3$ whenever $\delta < 3/2$.

Then Exact Triangle is in

$$\frac{n^3}{t^{1/3}} + \left(\left(\frac{n^3}{t^{1/3}} \cdot t^{1/3} \right)^{1-\varepsilon} = n^{2+\delta'/3} + n^{3-3\varepsilon} \right)$$

time, which is subcubic for any $\varepsilon > 0$ and $\delta < 3/2$.

Thus if Exact Triangle requires $n^{3-o(1)}$ time, as is implied by both the APSP and the 3SUM Hypotheses, then for every $\delta < 3/2$, listing $t = m^\delta$ triangles requires $m^{1-o(1)} t^{1/3}$ time.

If is easy to show that all triangles of an m edge graph can be listed in $O(m^{3/2})$ time. When a graph has $m^{3/2-o(1)}$ triangles, $m^{3/2-o(1)}$ time is also necessary since we need to write down the triangles. Hence, we have that for all $t \geq \Omega(m)$, the known running time for Sparse Triangle Listing (if $\omega = 2$) is conditionally optimal.

References

[1] Ilya Baran, Erik D. Demaine, Mihai Patrascu: Subquadratic Algorithms for 3SUM. Algorithmica 50(4): 584-596 (2008).

[2] Andreas Björklund, Rasmus Pagh, Virginia Vassilevska Williams, Uri Zwick: Listing Triangles. ICALP (1) 2014: 223–234.

[3] David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Patrascu, Perouz Taslakian: Necklaces, Convolutions, and X+Y. Algorithmica 69(2): 294-314 (2014).

[4] Timothy M. Chan, Yinzhan Xu: Simpler Reductions from Exact Triangle. SOSA 2024: 28-38.

[5] Mihai Patrascu: Towards polynomial lower bounds for dynamic problems. STOC 2010: 603–610.

[6] Virginia Vassilevska Williams, Ryan Williams: Finding, Minimizing, and Counting Weighted Subgraphs. SIAM J. Comput. 42(3): 831-854 (2013).

[7] Virginia Vassilevska Williams, Yinzhan Xu: Monochromatic Triangles, Triangle Listing and APSP. FOCS 2020: 786-797.